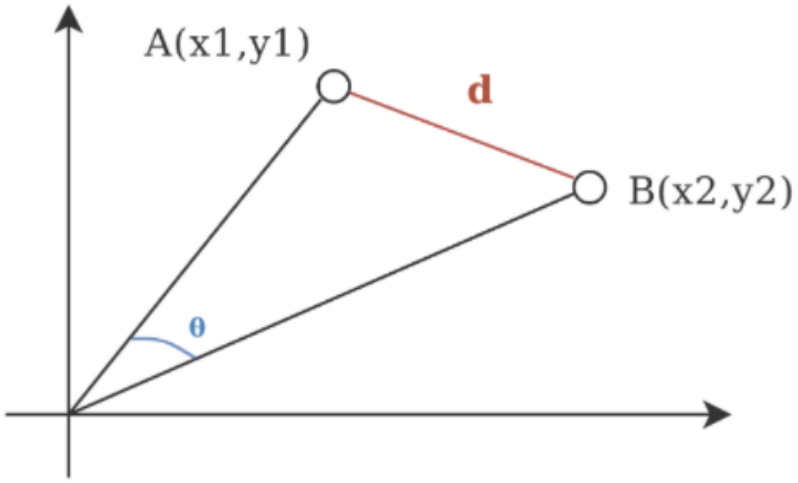
Efficient SIFT Feature Matching with Matrices

Recall the three steps to matching features between two images:

- Detection - Find a set of key points in an image
- Description - Create a vector description of the region around each key point
- Matching - Pair corresponding key points between two images

Now we will focus on the third step, matching. We can assume that we have already extracted a feature descriptor for the key points in each image. Using these descriptors, there are two common methods of finding the best matches between keypoints in two images. The first method, cosine similarity, uses the angle between two n-dimensional feature vectors to determine their similarity. For now, we will focus on the second, euclidean distance, which scores similarity using the magnitude of the difference between two vectors.



This image shows two 2D vectors, A and B. The euclidean distance between them is represented by d, and the cosine similarity is represented by θ.

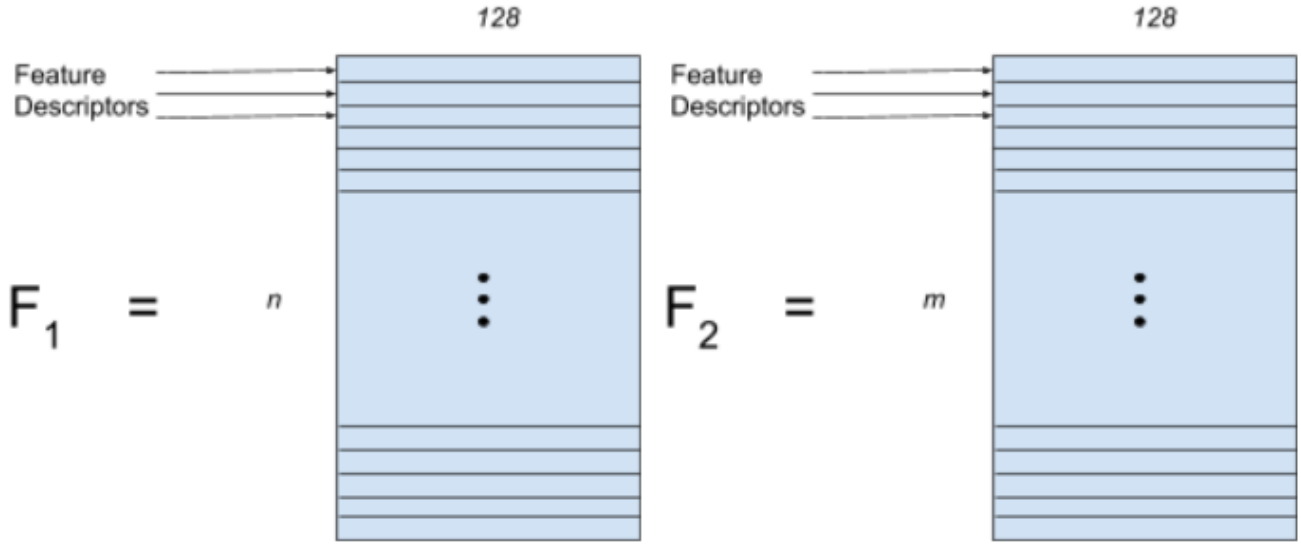
We can find the euclidean distance between two n-dimensional vectors, a and b, using the following equation:

$$||a - b|| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Using the formula above, we could calculate the euclidean distance by iterating (with a for loop) through the vectors a and b to get the sum of the square differences of each element. Then, to find the euclidean distances between each vector in one set and each vector in another set, we could add two more for loops to iterate through each set. If you were to actually implement this just using for loops, you would find that your code was uselessly slow. Fortunately for us though, the numpy library can perform a similar series of calculations much more quickly. To use numpy, we'll have to figure out how to do these computations using matrices instead of for loops. To start, we can rewrite the euclidean distance equation.

$$||a - b|| = \sqrt{(a - b)^2} = \sqrt{(a - b)(a - b)} = \sqrt{||a||^2 + ||b||^2 - 2ab}$$

Now, to represent our set of vectors, we can create two matrices, F_1 which has n vectors, and F_2 which has m vectors. Assuming our feature descriptors have a length of 128, these matrices will have shapes $(n, 128)$ and $(m, 128)$.



We can now use our equation for euclidean distance to create an $n \times m$ matrix that has the euclidean distance between every element in F_1 and every element in F_2. We will call this new matrix D.

$$D = \sqrt{||F_1||^2 + ||F_2||^2 - 2F_1F_2^T}$$

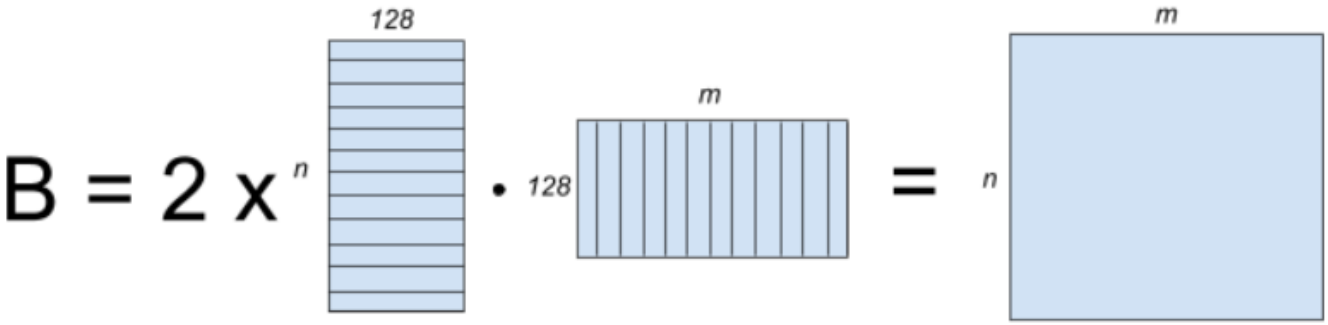
We will use the names A and B to help break this equation into smaller pieces.

$$D = \sqrt{A - B}$$

$$A = ||F_1||^2 + ||F_2||^2$$

$$B = 2F_1F_2^T$$

First we will focus on calculating B. Here we are creating an $n \times m$ matrix where each value is twice the dot product of one of the descriptors from F_1 and one of the descriptors from F_2. It is necessary to take the transpose of F_2 so that the descriptors are multiplied in the correct manner when the dot product is applied.

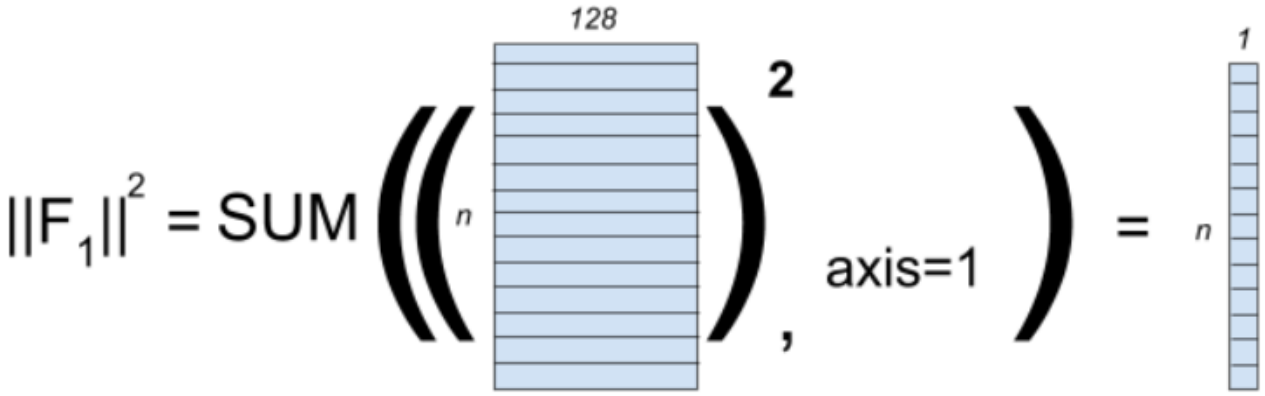


Next, we need to calculate A. We will start by just focusing on $||F_1||^2$. We will be able to calculate $||F_2||^2$ in the same manner and add it to $||F_1||^2$ to get A. Since F_1 is a matrix with n feature vectors, $||F_1||^2$ will be a matrix with shape $(n, 1)$ where each feature vector has been replaced by a single number, the square of its magnitude.

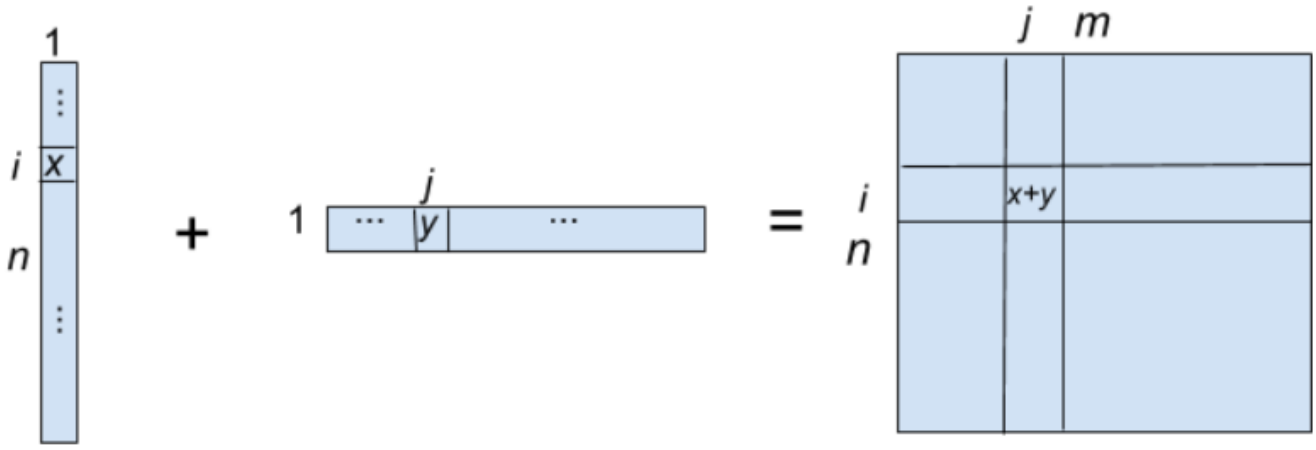
To find the square of the magnitude of a vector we can use the following equation:

$$||a||^2 = \sqrt{\sum_{i=1}^{n} a_i^2}^2 = \sum_{i=1}^{n} a_i^2$$

To do this calculation for all of the descriptors in F_1, we can multiply F_1 by itself element-wise to square each element in the matrix. Then, we can sum the matrix along the second dimension to create a sum for each feature descriptor. After computing this sum, it is important to make sure that the resulting matrix has a shape of $(n, 1)$ and not just (n).



Once we do the same thing for F_2, we will want to take the transpose of it so it has the shape $(1, m)$. Now using our two vectors with shapes $(n, 1)$ and $(1, m)$ we can perform what is known as a singleton expansion. This will yield a matrix with shape (n, m) where the value at position i, j is the sum of the ith value of the $||F_1||^2$ vector and the jth value of the $||F_2||^2$ vector. Using numpy, we are able to produce the singleton expansion by combining our two vectors using the '+' operator.



At this point we have two $n \times m$ matrices, A and B. Our A matrix has the sum of the squared feature descriptor magnitudes from F_1 and F_2, and our B matrix is twice the dot product of each pair of feature descriptors. We can now combine these to produce an $n \times m$ matrix, D, that contains the Euclidean distance between each feature descriptor in F_1 and each feature descriptor in F_2.

$$D = \sqrt{A - B}$$

Since we solved for D using matrices, we can make use of efficient numpy functions instead of using for loops! (Note: The euclidean distance isn't our final metric for determining what a good match is. Make sure to implement the ratio test in your solution!)